Stand: 5. Dezember 2000

# 4 Serverseitiges Scripting mit Perl



Stand: 5. September 2000

## 4.7 Ergänzungen

In diesem Abschnitt werden im wesentlichen Teile aus Kapitel 4 ergänzt, die dort nicht vollständig dargestellt wurden. Dies betrifft Operatoren, vordefinierte skalare Variable und vor allem die Standardfunktiionen.
Auch damit bleiben aber Themen, die im Kontext des Buches weniger oder keine Bedeutung haben, unbehandelt. Die *vollständige* Dokumentation von Perl samt aller Packages kann im Internet unter `http://www.perl.com/CPAN/` leicht beschafft werden.

## 4.7.1 Operatoren

| Rang | X | Operatoren | Beschreibung |
|---|---|---|---|
| 1 | 2 | `->` | Infix-Dereferenzierungsoperator |
| 2 | 1 | `++` | Um 1 erhöhen (Autoinkrementier) |
| | 1 | `--` | Um 1 erniedrigen (Autodekrementeier) |
| 3 | 2 | `**` | Potenzieren; **rechtsassoziativ!** |
| 4 | 1 | `\` | Referenz, z.B. `\$variable` |
| | | `! ~` | Negation und Einerkomplement |
| | | `+ -` | Unäres Plus und Minus |
| 5 | 2 | `=~` | Bindeoperator für die Suche nach Mustern |
| | | `!~` | dto. für die Suche nach nicht übereinstimmenden Mustern |
| 6 | 2 | `* / % x` | Multiplikation, Division, Restbildung, Wiederholung |
| 7 | 2 | `+ - .` | Addition, Subtraktion, Strinverkettung |
| 8 | 2 | `>> <<` | Bits nach rechts / nach links schieben |
| 9 | 2 | `< > <= >=` | kleiner, größer, kleiner gleich, größer gleich für Zahlen |
| | | `lt gt le ge` | kleiner, größer, kleiner gleich, größer gleich für Strings |
| 10 | 2 | `== != <=>` | gleich, ungleich, Größenvergleich für Zahlen |
| | | `eq ne cmp` | gleich, ungleich, alphabetischer Vergleich für Strings |
| | | | -1 für kleiner, 0 für gleich und 1 für größer |
| 11 | 2 | `&` | Bitweise UND |
| 12 | 2 | `\| ^` | Bitweise ODER und exkluxives ODER |
| 13 | 2 | `&&` | UND (Abbruch, wenn Operand links `==0` / `false`) |
| 14 | 2 | `\|\|` | ODER (Abbruch, wenn Operand links `true` / `!= 0`) |
| 15 | 2 | `.. ...` | Bereichsoperator für Aufzählungen |
| 16 | 3 | `?:` | Bedingungsoperator: *bedingung* ? *truewert* : *falsewert* |
| 17 | 2 | `= += -=` etc. | Zuweisungsoperatoren; **rechtsassoziativ!** |
| 18 | 2 | `,` | Trenner für Listenelemente |
| | | `=>C` | Trenner in Textschlüssel-/Wertepaaren für assoziative Arrays |
| 19 | 1 | `not` | logisches NICHT mit geringem Vorrang |
| 20 | 2 | `and` | logisches AND mit geringem Vorrang |
| 21 | 2 | `or xor` | logische OR und XOR mit geringem Vorrang |

In der ersten Spalte ist der Rang der Operatoren angegeben. Z.B. wird in dem Ausdruck `$a + $b * $c` erst die Multiplikation (Rang 6) und dann die Addition (Rang 7) ausgeführt, oder in $ b = $a == 5 erst die Prüfung auf Gleichheit (Rang 10) und dann die Zuweisung (Rang 17). Rechtsassoziativ sagt aus, dass bei aufeinanderfolgenden *gleichrangigen* Operatoren erst der rechtsstehende Operator zum Zuge keommt, dann der weiter links stehende etc. Die Mehrfachzuweisung `$a = $b = $c` entspricht also `$a = ($b = $c)`, denn das Zuweisungszeichen ist rechtassoziativ. Dagegen entspricht `$a + $b - $c` der Klammerung `($a + $b) - $c`, denn + und - sind gleichrangig und linksassoziativ.

## 4.7.2 Vordefinierte skalare Variable

Die zusätzlich angegebenen Langnamen, beispielsweise `$PREMATCH` zu `$`` und `$MATCH` zu `$&`, sind nur in Verbindung mit

```
use English
```

verwendbar.

### Lokale Variable

**`$&` `$MATCH`**
Bei Mustersuche der letzte erkannte String (Treffer).

**`$`` `$PREMATCH`**
Vor dem Treffer stehender String.

**`$´` `$POSTNMATCH`**
Nach dem Treffer stehender String.

**`$zahl` `$1, $2, ...`**
Mittels Klammernpaaren ausgeschnitteneTeilmuster aus einem erkannten Muster.

**`$+` `$LAST_PAREN_MATCH`**
Die letzte im zuletzt erkannten Suchmuster darin erkannte Klammer.

### kontextabhängige Variable

**`$%` `$FORMAT_PAGE_NUMBER`**
Aktuelle Seitennummer des gewählten Ausgabekanals.

**`$=` `$FORMAT_LINES_PER_PAGE`**
Anzahl der Zeilen pro Seite; voreingestellt sind 60 Zeilen).

**`$-` `$FORMAT_LINES_LEFT`**
Anzahl der noch verbleibenden Zeilen auf einer Seite.

**`$~` `$FORMAT_NAME`**
Formatierte Ausgabe: Formatname.

**`$^` `$FORMAT_TOP_NAME`**
Formatierte Ausgabe: Name des Formatkopfes

**`$|` `$OUTPUT_AUTOFLUSH`**
Wenn ungleich `0`, so wird nach jedem `write` oder `print` der aktuelle Ausgabekanal (z.B. `STDOUT`) der Ausgabepuffer geleert. Voreingestallt ist `0`.

**`$ARGV`**
Enthält den Namen der aktuellen Datei, wenn über `<HANDLE>` gelesen wird.

## Globale skalare Variable

**$_ $ARG**
Standardvariable für Eingaben, Ausgaben, Funktionsparameter und Mustererkennung.

**$. $NR, $INPUT_LINE_NUMBER**
Die Nummer der aktuellen eingelesenen Eingabezeile in Bezug auf den zuletzt gelesenen Handle.

**$/ $RS, $INPUT_RECORD_SEPARATOR**
Trennzeichen in Eingabedaten. (Voreingestellt ist `\n`.)

**$, $OFS, $OUTPUT_FIELD_SEPARATOR**
String (normalerweise leer), der bei der Ausgabe einer Liste mittels `print` zwischen die Listenelemente geschoben wird. Beispiel:

```
$, = "%";
print "a", "b", "c";  // -> a%b%c
```

**$" $LIST_SEPARATOR**
Ähnlich `$,`, aber für in Stringliteralen eingestellte Arrays, z.B.

```
$" = "?"; @ar = (2, 4, 8);
print "dieses @ar ausgeben";   // -> dieses 2?4?8 ausgeben
```

**$\ $ORS, $OUTPUT_RECORD_SEPARATOR**
Sofern `$\` einen String enthält (normalerweise ist er leer), wird er bei allen Ausgaben mittels `print` abschließend ausgedruckt.

**$# veraltet**
Ausgabeformat für Zahlen (`printf` verwenden)

**$* veraltet**
„Multiline Match“ aktivieren; siehe `m`- und `s`-Modifizierer

**$? $CHILD_ERROR**
Status des zuletzt ausgeführten `` `...` ``-Befehls.

**$] $PERL_VERSION**
Perl Versionsnummer, z.B. 5.005.

**$[ veraltet!**
Der Index des ersten Elkementes in einem Array oder einer Liste und das erste Zeichen in einem String. Die Voreinstellung ist 0.

**$; $SUBSEP, $SUBSCRIPT_SEPARATOR**
Trennsymbol für die Emulation mehrdimensionaler Listen.

**$! $ERRNO, $OS_ERROR**
Aktuelle Fehlernummer oder Fehlerstring.

**$@ $EVAL_ERROR**
Perl-Fehlermeldung der letzten `eval()`- oder `do ausdruck`-Ausführung.

**$:** **$FORMAT_LINE_BREAK_CHARACTERS**
Eine Menge von Zeichen, nach denen ein String umgebrochen werden kann

**$0** **$PROGRAM_NAME**
Der Name der Datei, die das Perlprogramm enthält.

**$$** **$PID, $PROCESS_ID**
Prozess-ID des Perl-Interpreters, des garade das Scriptprogramm ausführt.

Die Standardfunktionen sind noch in Arbeit.